Task:

To find the best vectorizer for SVC

In [18]:

```
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

# using the SQLite Table to read data.
con = sqlite3.connect('database.sqlite')

#filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 """, con)

# Give reviews with Score>3 a positive rating, and reviews with a score<3 a negative rating.
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print(filtered_data.shape)
```

```
(525814, 10)
```

In [19]:

```
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape
```

Out[19]:

```
(364173, 10)
```

In [20]:

```
final.head(2)
```

Out[20]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Su
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	939340800	E educ
138688	150506	0006641040	A2IW4PEEKO2R0U	Tracy	1	1	1	1194739200	L boo th '

In [21]:

```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
#Before starting the next phase of preprocessing lets see the number of entries left
print(final.shape)

#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(364171, 10)
```

Out[21]:

```
1    307061
0     57110
Name: Score, dtype: int64
```

In [22]:

```
final.sort_values('Time',axis=0,ascending=True,inplace=True,kind='quicksort')
final.head(2)
```

Out[22]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summ
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	939340800	EV bo educati
138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2	1	940809600	This w seri great to sp time

In [23]:

```
stop = set(stopwords.words('english')) #set of stopwords
sno = nltk.stem.SnowballStemmer('english') #initialising the snowball stemmer

def cleanhtml(sentence): #function to clean the word of any html-tags
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', sentence)
    return cleantext
def cleanpunc(sentence): #function to clean the word of any punctuation or special characters
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned
```

In [26]:

```
#Code for implementing step-by-step the checks mentioned in the pre-processing phase
# this code takes a while to run as it needs to run on 500k sentences.
i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
for sent in final['Text'].values:
    filtered_sentence=[]
    #print(sent);
    sent=cleanhtml(sent) # remove HTMl tags
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(sno.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                    if (final['Score'].values)[i] == 1:
                        all_positive_words.append(s) #list of all words used to describe positive r
eviews
                    if(final['Score'].values)[i] == 0:
                        all_negative_words.append(s) #list of all words used to describe negative r
eviews reviews
                else:
                    continue
            else:
                continue
    #print(filtered_sentence)
    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    #print("***********************************************************************")

    final_string.append(str1)
    i+=1
```

In [27]:

```
final['CleanedText']=final_string #adding a column of CleanedText which displays the data after pr
e-processing of the review
final['CleanedText']=final['CleanedText'].str.decode("utf-8")
final.head(3)
```

Out[27]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Su
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	939340800	E educ
138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2	1	940809600	This s great spe
417839	451856	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	0	0	1	944092800	Enter

In [ ]:

```
import pickle
pickle.dump(final, open('final.p', 'wb'))
#final_sent = pickle.load(open('data.p','rb'))
final.shape
```

In [66]:

```
import pickle
final = pickle.load(open('final.p','rb'))


from sklearn.model_selection import train_test_split
##Sorting data according to Time in ascending order for Time Based Splitting
time_sorted_data = final.sort_values('Time', axis=0, ascending=True, inplace=False, kind='quicksort
', na_position='last')
final.head(2)
```

Out[66]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summ
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	939340800	EVI bo educati
138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2	1	940809600	This w seri great to sp time

In [67]:

```
# sampling the data for ease
y = final['Score'].sample(frac=.29)
x = final['CleanedText'].sample(frac=.29)
x.shape,y.shape
```

Out[67]:

```
((105610,), (105610,))
```

BOW

In [26]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

X_tra, X_tes, y_train, y_test = train_test_split(x.values,y.values,test_size=0.3,random_state=0)

#Implementing BAG of words
bow = CountVectorizer(ngram_range=(0,1))
X_tf =bow.fit_transform(X_tra)

# Standerdising the data
norm = StandardScaler(with_mean = False)
X_train = norm.fit_transform(X_tf)

# tfidf test
X_tfte = bow.transform(X_tes)

# Standerdising the data
X_test = norm.transform(X_tfte)



X_train.shape,y_train.shape,X_test.shape,y_test.shape
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:590:
DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

Out[26]:

```
((73927, 32962), (73927,), (31683, 32962), (31683,))
```

In [ ]:

```

```

In [27]:

```python
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

param_grid = {'alpha':[0.0005,0.0001,0.005,0.001,0.05,0.01,0.1]}
clf = SGDClassifier()
gsv = GridSearchCV(clf,param_grid,cv=3,scoring="f1")
gsv.fit(X_train,y_train)

print("Best HyperParameter: ",gsv.best_params_)
print("Best f1: %.2f%%"%(gsv.best_score_*100))
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

```
Best HyperParameter:  {'alpha': 0.1}
Best f1: 89.76%
```

In [28]:

```python
model = linear_model.SGDClassifier(alpha=0.1)
model.fit(X_train,y_train)
y_pred = model.predict(X_train)
```

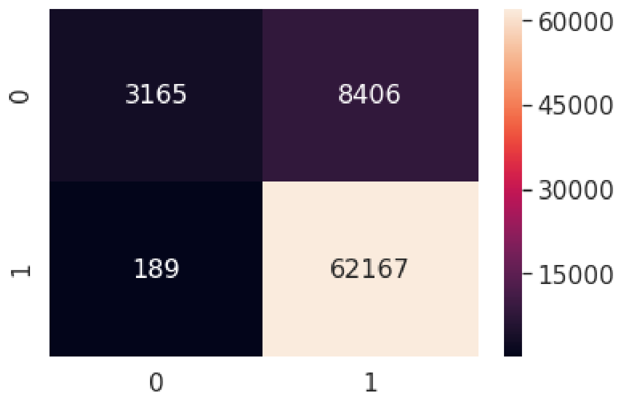
In [30]:

```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

print("Accuracy on the set: %0.3f%%"%(accuracy_score(y_train, y_pred1)*100))
print("F1-Score on the set: %0.3f%%"%(f1_score(y_train, y_pred1)*100))
print("Precision_score on test set: %0.3f%%"%(precision_score(y_train, y_pred1)*100))
print("Recall_score on test set: %0.3f%%"%(recall_score(y_train, y_pred1)*100))
print("Confusion Matrix of test set:\n [ [TN  FP]\n [FN TP] ]\n")
print(result)
sns.set(font_scale=1.4)#for label size
sns.heatmap(result, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Accuracy on the set: 74.427%
F1-Score on the set: 84.959%
Precision_score on test set: 84.301%
Recall_score on test set: 85.628%
Confusion Matrix of test set:
 [ [TN  FP]
 [FN TP] ]

[[ 3165  8406]
 [  180 62167]]
[  189 62167]]
```

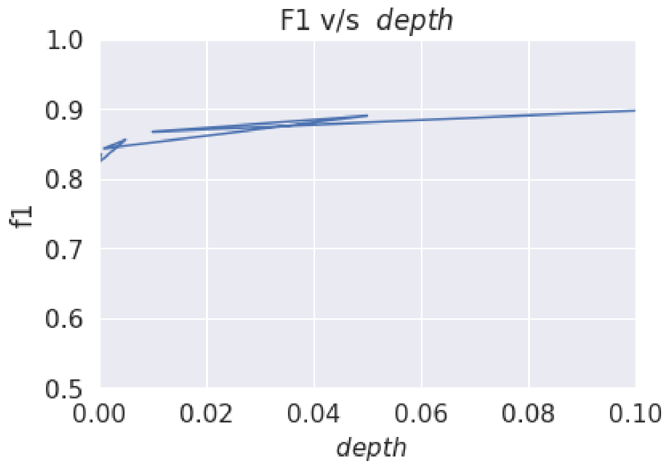
Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32389a14e0>
```



In [31]:

```python
# plotted for my understanding only. ignore plot
import matplotlib.pyplot as plt
score =gsv.cv_results_
x=[]
y=[]

for a in score['param_alpha']:
    x.append(a)
for a in score['mean_test_score']:
    y.append(a)

plt.xlim(0.0001,0.1)
plt.ylim(0.5,1)
plt.xlabel(r"$\ depth $",fontsize=15)
plt.ylabel("f1")
plt.title(r'F1 v/s $\ depth$')
plt.plot(x,y)
plt.show()
```



In [ ]:

```

```

TFIDF

In [46]:

```python
# sampling the data for ease
y = final['Score'].sample(frac=.29)
x = final['CleanedText'].sample(frac=.29)
```

```
x.shape,y.shape
```

Out[46]:

```
((105610,), (105610,))
```

In [47]:

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

X_tra, X_tes, y_train, y_test = train_test_split(x.values,y.values,test_size=0.3,shuffle=False)

#Implementing BAG of words
tfidf = TfidfVectorizer(ngram_range=(0,1),dtype=float)
X_tf =tfidf.fit_transform(X_tra)

# Standerdising the data
norm = StandardScaler(with_mean = False)
X_train = norm.fit_transform(X_tf)

# tfidf test
X_tfte = tfidf.transform(X_tes)

# Standerdising the data
X_test = norm.transform(X_tfte)


from sklearn.model_selection import TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=4)
for train, cv in tscv.split(X_train):
    print(X_train[train].shape, X_train[cv].shape)
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/feature_extraction/text.py:1547:
UserWarning: Only (<class 'numpy.float64'>, <class 'numpy.float32'>, <class 'numpy.float16'>) 'dty
pe' should be used. <class 'float'> 'dtype' will be converted to np.float64.
  UserWarning)
```

```
(14787, 33227) (14785, 33227)
(29572, 33227) (14785, 33227)
(44357, 33227) (14785, 33227)
(59142, 33227) (14785, 33227)
```

In [48]:

```python
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

param_grid = {'alpha':[0.0005,0.0001,0.005,0.001,0.05,0.01,0.1]}
clf = SGDClassifier()
gsv2 = GridSearchCV(clf,param_grid,cv=3,scoring="f1")
gsv2.fit(X_train,y_train)

print("Best HyperParameter: ",gsv2.best_params_)
print("Best f1: %.2f%%"%(gsv2.best_score_*100))
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

```
Best HyperParameter:  {'alpha': 0.1}
Best f1: 89.69%
```

In [49]:

```
model2 = linear_model.SGDClassifier(alpha = 0.1)
model2.fit(X_train,y_train)
y_pred2 = model2.predict(X_train)
model2
```

Out[49]:

```
SGDClassifier(alpha=0.1, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

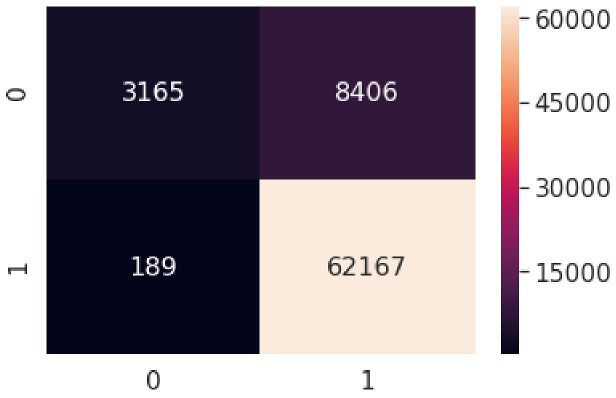
In [50]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

print("Accuracy on the set: %0.3f%%"%(accuracy_score(y_train, y_pred2)*100))
print("F1-Score on the set: %0.3f%%"%(f1_score(y_train, y_pred2)*100))
print("Precision_score on test set: %0.3f%%"%(precision_score(y_train, y_pred2)*100))
print("Recall_score on test set: %0.3f%%"%(recall_score(y_train, y_pred2)*100))

print("Confusion Matrix of test set:\n [ [TN  FP]\n [FN TP] ]\n")
print(result)
sns.set(font_scale=1.4)#for label size
sns.heatmap(result, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Accuracy on the set: 88.667%
F1-Score on the set: 93.692%
Precision_score on test set: 88.330%
Recall_score on test set: 99.747%
Confusion Matrix of test set:
 [ [TN  FP]
 [FN TP] ]

[[ 3165  8406]
 [  189 62167]]
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32136231d0>
```



Avg w2c

In [53]:

```
sent_of_train=[]
```

```
for sent in X_tra:
    sent_of_train.append(sent.split())
```

In [54]:

```
#word to vector
from gensim.models import Word2Vec
w2v_model=Word2Vec(sent_of_train,min_count=3,size=200, workers=4)# words which occurs 3 times; 500
dimensions

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 3 times ",len(w2v_words))
```

```
number of words that occured minimum 3 times  14531
```

In [55]:

```
# compute average word2vec for each review for X_train .
from tqdm import tqdm
import numpy as np

train_vectors = []
for sent in tqdm(sent_of_train):
    sent_vec = np.zeros(200)
    cnt_words =0;
    for word in sent:
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    train_vectors.append(sent_vec)
```

```
100%|██████████| 73927/73927 [14:36<00:00, 84.33it/s]
```

In [59]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import TimeSeriesSplit

# Data-preprocessing: Standardizing the data
sc = StandardScaler(with_mean = False)
X_train3 = sc.fit_transform(train_vectors)

tscv = TimeSeriesSplit(n_splits=4)


y_train.shape,X_train3.shape
```

Out[59]:

```
((73927,), (73927, 200))
```

In [60]:

```
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

param_grid = {'alpha':[0.0005,0.0001,0.005,0.001,0.05,0.01,0.1]}
clf = SGDClassifier()
gsv3 = GridSearchCV(clf,param_grid,cv=3,scoring="f1")
gsv3.fit(X_train3,y_train)

print("Best HyperParameter: ",gsv3.best_params_)
print("Best f1: %.2f%%"%(gsv3.best_score_*100))
print('Best f1: %.2f%%' %(gsv3.best_score_*100))
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

```
Best HyperParameter:  {'alpha': 0.05}
Best f1: 91.53%
```

In [63]:

```
model3 = linear_model.SGDClassifier(alpha = 0.05)
model3.fit(X_train3,y_train)
y_pred3 = model3.predict(X_train3)
model3
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

Out[63]:

```
SGDClassifier(alpha=0.05, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

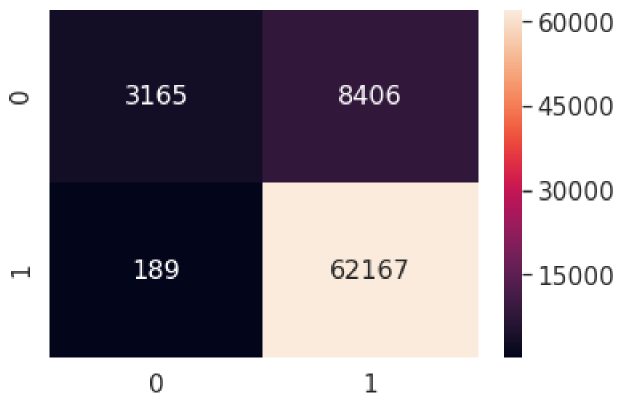
In [65]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

print("Accuracy on the set: %0.3f%%"%(accuracy_score(y_train, y_pred3)*100))
print("F1-Score on the set: %0.3f%%"%(f1_score(y_train, y_pred3)*100))
print("Precision_score on test set: %0.3f%%"%(precision_score(y_train, y_pred3)*100))
print("Recall_score on test set: %0.3f%%"%(recall_score(y_train, y_pred3)*100))

print("Confusion Matrix of test set:\n [ [TN  FP]\n [FN TP] ]\n")
print(result)
sns.set(font_scale=1.4)#for label size
sns.heatmap(result, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Accuracy on the set: 84.376%
F1-Score on the set: 91.526%
Precision_score on test set: 84.376%
Recall_score on test set: 100.000%
Confusion Matrix of test set:
 [ [TN  FP]
 [FN TP] ]

[[ 3165  8406]
 [  189 62167]]
```

Out[65]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f31f56e74a8>
```



AVG TF-IDF W2V

In [68]:

```
# sampling the data for ease
y = final['Score'].sample(frac=.29)
x = final['CleanedText'].sample(frac=.29)
x.shape,y.shape
```

Out[68]:

```
((105610,), (105610,))
```

In [69]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

X_tra, X_tes, y_train, y_test = train_test_split(x.values,y.values,test_size=0.3,random_state=0)
```

In [70]:

```
sent_of_train=[]
for sent in X_tra:
    sent_of_train.append(sent.split())
```

In [71]:

```
#word to vector
from gensim.models import Word2Vec
w2v_model=Word2Vec(sent_of_train,min_count=3,size=200, workers=4)# words which occurs 3 times; 500
dimensions

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 3 times ",len(w2v_words))
```

```
number of words that occured minimum 3 times  14409
```

In [72]:

```
# TF-IDF weighted Word2Vec
tf_idf_vect = TfidfVectorizer()

# final_tf_idf1 is the sparse matrix with row= sentence, col=word and cell_val = tfidf
final_tf_idf1 = tf_idf_vect.fit_transform(X_tra)
```

In [101]:

```
# tfidf words/col-names
tfidf_feat = tf_idf_vect.get_feature_names()

# compute TFIDF Weighted Word2Vec for each review for X_test .
tfidf_test_vectors = [];
row=0;
for sent in tqdm(sent_of_train):
    sent_vec = np.zeros(200)
    weight_sum =0;
    for word in sent:
        if word in w2v_words:
            vec = w2v_model.wv[word]
            # obtain the tf_idfidf of a word in a sentence/review
            tf_idf = final_tf_idf1[row, tfidf_feat.index(word)]
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
    if weight_sum != 0:
        sent_vec /= weight_sum
    tfidf_test_vectors.append(sent_vec)
    row += 1
```

```
 61%|██████    | 44952/73927 [38:58<48:34,  9.94it/s]  IOPub message rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_msg_rate_limit`.

Current values:
NotebookApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
NotebookApp.rate_limit_window=3.0 (secs)

100%|██████████| 73927/73927 [1:04:43<00:00, 19.04it/s]
```

In [113]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import TimeSeriesSplit

# Data-preprocessing: Standardizing the data
sc = StandardScaler(with_mean = False)
X_train1 = sc.fit_transform(tfidf_test_vectors)
```

In [114]:

```
model = linear_model.SGDClassifier()
model.fit(X_train1,y_train)
y_pred = model.predict(X_train1)
model
```

```
/home/ash_sa8/.local/lib/python3.5/site-packages/sklearn/linear_model/stochastic_gradient.py:144:
FutureWarning: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are l
```

```
eft unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_i
ter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

Out[114]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

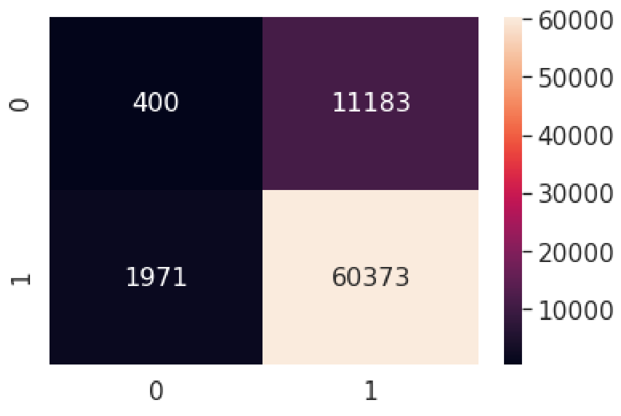
In [116]:

```
print("Accuracy on the set: %0.3f%%"%(accuracy_score(y_train, y_pred)*100))
print("F1-Score on the set: %0.3f%%"%(f1_score(y_train, y_pred)*100))
print("Precision_score on test set: %0.3f%%"%(precision_score(y_train, y_pred)*100))
print("Recall_score on test set: %0.3f%%"%(recall_score(y_train, y_pred)*100))

print("Confusion Matrix of test set:\n [ [TN  FP]\n [FN TP] ]\n")
print(result)
sns.set(font_scale=1.4)#for label size
sns.heatmap(result, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Accuracy on the set: 82.207%
F1-Score on the set: 90.176%
Precision_score on test set: 84.372%
Recall_score on test set: 96.839%
Confusion Matrix of test set:
 [ [TN  FP]
 [FN TP] ]

[[  400 11183]
 [ 1971 60373]]
```

Out[116]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbcca8bc630>
```



From the above model, Prefer to choose the best model with high precision and recall rate

Precision = TP / TP + FP	Recall = TP/P
1. BOW	
84.30 %	85.62%
2. TFIDF	
88.33 %	99.74%
3. AVG-W2V	
84.37 %	100%
4. TFIDF-W2V	
84.37 %	96.83%

We choose TFIDF model for SVC implementation